10/511187
WO 03/088605 PCT/IT02/00430
DT04 Rec'd PCT/PTO 12 OCT 2004
VPRTS

SCHEDULING A SHARED RESOURCE AMONG SYNCHRONOUS AND ASYNCHRONOUS PACKET FLOWS

TECHNIQUE SECTOR

This invention refers to the packet communication systems, and in particular to the scheduling criteria of a shared resource, i.e. the criteria used to select the packet to which the resource is to be assigned each time this occurs.

The solution given in the invention has been developed both for radio resource scheduling (e.g.: MAC or Medium Access Control level scheduling), and for the scheduling of computational and transmissive resources in the network nodes, for example, for flow scheduling with different service quality on Internet Protocol router (IP). The following description is based especially on the latter application example, and is given purely as an example and does not limit the scope of the invention.

INTRODUCTION

For several years now, the widespread application and rapid evolution of the packet networks have given rise to the problem of integrating the traditional services offered by the old generation packet networks (electronic mail, web surfing, etc.) and the new services previously reserved for circuit switching networks (real-time video, telephony, etc.) into the so-called integrated services networks.

Systems like UMTS, for example, for which a fixed packet network component (core network) is envisaged, must simultaneously handle voice and data services, and offer support for the development of new services be they real-time or not.

The integrated services networks must therefore be able to handle traffic flows with different characteristics and to offer each type of flow a suitable service quality, a set of performance indexes negotiated between user and service

provider, which must be guaranteed within the terms agreed upon.

One of the key elements in providing the service quality requested is the scheduling system implemented on the network nodes, i.e. the system used to select the packet to be transmitted from those present on the node; this system must obviously embody contrasting characteristics like flexibility, in terms of capacity to provide different types of services, simplicity, a characteristic that makes it possible to use in environments that require high transmission speeds and the handling of numerous transmission flows, and efficiency in the use of the shared resource (e.g. the transmissive means).

The need to guarantee a given level of service quality (or Qos) in the packet networks is constantly increasing, as can be seen for example in the documents US-A-6 091 709, US-A-6 147 970 or EP-A-1 035 751.

This invention in fact is the development of the solution described in the industrial invention patent request TO2000A001000 and in the corresponding request PCT/IT 01/00536.

The previous solution basically applies to the scheduling of a service resource shared between several information packet flows in which the flows generate respective associated queues and are serviced when the server gives permission to transmit.

The flows are divided into synchronous flows, which require a minimum service rate guarantee, and into asynchronous flows, which use the service capacity of the resource that is left unused by the synchronous flows. The solution in question includes the following:

- provides a server that visits the queues associated with the flows in successive cycles, granting each queue a target token rotation time (or "revolution"), called TTRT,

which identifies the time required for the server to complete the queue visiting cycle,

- associates each synchronous flow with a synchronous capacity value indicating the maximum time the synchronous flow can be serviced before its transmission permission is revoked by the server,

- associates each asynchronous flow with a first *lateness(i) value,* indicating the delay that must be made up for the respective queue to have the right to be serviced, plus another value (*last_token_time*) indicating the moment the server visited the respective queue in the previous cycle, which determines the time elapsed since the server's previous visit,

- services each queue associated to a synchronous flow for a maximum period of time equal to the above-mentioned synchronous capacity value, and

- services each queue associated to an asynchronous flow only if the server's visit occurs before the expected moment. This advance is obtained from the difference between the aforesaid TTRT time and the time that has elapsed since the server's previous visit and the accumulated delay.

If this difference is positive it defines the maximum service time for each queue associated to an asynchronous flow.

The solution referred to above has proved to be completely satisfactory from an operational point of view. The experience gained by the "Petitioner" has however shown that the solution can be further developed and improved as illustrated in this invention.

This applies particularly to the following aspects:

- the possibility of offering different types of service while keeping computational costs low: an important feature for computer network applications that must guarantee service quality for its users, like the IP networks with Intserv

(Integrated Services, as per IETF specification) or Diffserv (Differentiated Integrated Services, as per IETF specification), or for the radio resource scheduling systems like the MAC level scheduling algorithms (W-LAN systems, third generation radio-mobile services);

- the possibility of guaranteeing the bit rate of the various flows, the maximum queuing delay and the maximum occupation of the buffers of each flow for synchronous traffic;

- flexibility, in terms of capacity to provide two different types of services at the same time, rate-guaranteed (suitable for synchronous flows) and fair queuing (suitable for asynchronous flows), especially in service integration networks;

- the possibility of isolating transmission flows, i.e. it makes the service offered to a single flow independent from the presence and behaviour of other flows;

- low computational complexity in terms of the number of operations necessary to select the packet to be transmitted; this feature makes it possible to use in environments that require high transmission speeds and the handling of numerous transmission flows, also in view of a possible implementation in hardware;

- adaptability, in the sense that it can handle a change in the operating parameters (e.g. the number of flows present) by redistributing its resources without having to resort to complex procedures; and

- analytic describability, i.e. it gives a complete analytic description of the system's behaviour, which makes it possible to relate the service quality measurements to the system parameters.

Another important aspect is equity, i.e. the possibility to manage in the same way both the transmission flows that receive a rate-guaranteed service, and those that receive a

fair-queuing service, giving each one a level of service that is proportional to that requested, even in the presence of packets of different lengths.

DESCRIPTION OF THE INVENTION

The aim of this invention is to develop even further the already known solution referred to previously with special attention to the aforesaid aspects.

According to this invention, this aim can be reached by using a scheduling procedure having the characteristics referred to specifically in the following claims.

The invention also refers to the relative system.

Briefly, the solution given in the invention operates a scheduling system that can be defined with the name introduced in this patent request – Packet Timed Token Service Discipline or PTTSD.

At the moment, this scheduling system is designed to work on a packet-computer network switching node and is able to multiplex a single transmission channel into several transmission flows.

The system offers two different types of service: rate-guaranteed service, suitable for transmission flows (henceforth, "synchronous flows") that require a guaranteed minimum service rate, and a fair-queueing service, suitable for transmission flows (henceforth "asynchronous flows") that do not require any guarantee on the minimum service rate, but which benefit from the greater transmission capacity available. The system provides the latter, however, with an equal sharing of the transmission capacity not used by the synchronous flows.

The traffic from each transmission flow input on the node is inserted in its own queue (synchronous or asynchronous queues) from which it will be taken to be transmitted. The server visits the queues in a fixed cyclic

order and grants each queue a service time established according to precise timing constraints at each visit.

The server initially visits the synchronous queues twice during a revolution, thus completing a major cycle and a minor or recovery cycle, and then moves on to visit the asynchronous queues.

BRIEF DESCRIPTION OF THE FIGURE

The following description of the invention is given as a non-limiting example, with reference to the annexed drawing, which includes a single block diagram figure that illustrates the operating criteria of a system working according to the invention.

DESCRIPTION OF A PREFERRED FORM OF EXECUTION

A scheduling system as given in the invention is able to multiplex a single transmission channel into several transmission flows.

The system offers two different types of service: a rate-guaranteed service, suitable for transmission flows (henceforth i synchronous flows where $i = 1, 2, \ldots, N_S$) that require a guaranteed minimum service rate, and a best-effort service, suitable for transmission flows (henceforth j asynchronous flows where $j = 1, 2, \ldots, N_A$) that do not require any guarantee on the service rate. The system provides the latter, however, with an equal sharing of the transmission capacity not used by the synchronous flows.

It should be supposed that N_S and N_A are non-negative integers and that each synchronous flow $i=1..N_s$ requires a service rate equal to r_i , and that the sum of the service rates requested by the synchronous flow does not exceed the capacity of channel C ($\sum_{i=1}^{N_S} r_i \leq C$).

The traffic from each transmission flow input on the node is inserted in its own queue (synchronous or asynchronous queues will be discussed later) from which it

will be taken to be transmitted. The server 10 visits the queues in a fixed cyclic order (ideally illustrated in the figure of the drawings with trajectory T and arrow A), granting each queue a service time established according to precise timing constraints at each visit.

The procedure referred to in the invention includes an initialisation stage followed by cyclic visits to the queues. These procedures will be discussed below.

Initialisation

First of all, it is necessary to give the system the information relating to the working conditions: how many synchronous flows there are (in general: N_S), what the transmission rate requested by each synchronous flow is, how many asynchronous flows there are, the target rotation time (TTRT), i.e. how long a complete cycle during which the sever visits all the queues once is to last.

Synchronous flows

Each synchronous flow i, $i = 1..N_s$, is associated, according to an appropriate allocation policy, to a variable H_i (synchronous capacity), which measures the maximum time for which the traffic of a synchronous flow can be transmitted before the server takes the transmission permission away. The possible allocation policies will be described below. A variable Δ_i, initially nil, is associated to each synchronous flow, and stores the amount of transmission time available to the flow.

Asynchronous flows

Each asynchronous flow j, $j = 1..N_A$, is associated with two variables, L_j and *last_visit_time*$_j$; the first variable stores the delay or lag that must be made up for the asynchronous queue j to have the right to be serviced; the second variable stores the instant the server visited the

asynchronous queue j in the previous cycle. These variables are respectively initialised to zero and to the instant the revolution in progress when the flow is activated started.

This way of proceeding means that the asynchronous flows can be activated at any moment, not necessarily at system startup.

<u>Visit to a generic synchronous queue i, with i = 1...N_S during the major cycle</u>

A synchronous queue can be serviced for a period of time equal to the maximum value of the variable Δ_i. This variable is incremented by H_i (value decided during initialisation) when the queue is visited in the major cycle, and decremented by the transmission time of each packet transmitted.

The service of a queue during the major cycle ends when either the queue is empty (in which case the variable Δ_i is reset), or the time available (represented by the current value of Δ_i) is not sufficient to transmit the packet that is at the front of the queue.

<u>Visit to a generic synchronous queue i, i = 1...N_S during the minor cycle</u>

During the minor (or recovery) cycle a synchronous queue can transmit only one packet, provided the variable Δ_i has a strictly positive value. If transmission takes place, the variable Δ_i is decremented by the transmission time.

<u>Visit to a generic asynchronous queue j, with j =1,...,N_A</u>

An asynchronous queue can only be serviced if the server's visit takes place before the expected instant. To calculate whether the server's visit is in advance, subtract the time that has elapsed since the previous visit and the accumulated delay L_j from the target rotation time $TTRT$.

If this difference is positive, it is the period of time for which the asynchronous queue j has the right to be serviced, and in this case the variable L_j is reset.

If the difference is negative, the server is late and the queue j cannot be serviced; in this case the delay is stored in the variable L_j. The asynchronous queue service ends when the queue is empty, or the time available (which is decremented each time a packet is transmitted) is not sufficient to transmit the packet that is at the front of the queue.

Visit sequence during a revolution

A double scan is made on all the synchronous queues (major and minor cycles) during one revolution, and then the asynchronous queues are visited. The minor cycle ends the moment one of the following events takes place:

- the last synchronous queue has been visited;
- a period of time that is equal to or greater than the sum of the capacity of all the synchronous queues has elapsed since the beginning of the major cycle.

Analytic guarantees

The synchronous capacities are linked to the target rotation time $TTRT$ and to the duration of the transmission of the longest packet $\tau_{\max}$ by the following inequality, which must always be verified:

$$\sum_{i=1}^{N_S} H_i + \tau_{\max} \leq TTRT \quad (1)$$

Minimum transmission rate for synchronous flows

In hypothesis (1), the system as illustrated herein guarantees that the following normalised transmission rate will be guaranteed for each synchronous flow:

$$\gamma_i = \frac{N_A + 1}{N_A + \sum_{h=1}^{N_S} X_h + \alpha} \cdot X_i$$

with:

$X_i = H_i / TTRT$

$\alpha = \tau_{max} / TTRT$

and it is also possible to guarantee that, given any period of time $[t_1, t_2)$ in which the generic synchronous queue i is never empty, the service time $W_i(t_1, t_2)$ received from the queue i in $[t_1, t_2)$ verifies the following inequality:

$$\gamma_i \cdot (t_2 - t_1) - W_i(t_1, t_2) \le \Lambda_i < \infty \quad (2)$$

where:

$$\Lambda_i = \begin{cases} H_i \cdot (2 - \gamma_i) + (1 + \gamma_i) \cdot \tau_i & se\ H_i \ge \tau_i \\ \tau_i + 2 \cdot H_i & se\ H_i < \tau_i \end{cases}$$

and τ_i is the transmission time of the longest packet for the flow i.

Expression (2) seen previously establishes that the service supplied by the i synchronous flow system of the type described here does not differ by more than Λ_i from the service that the same flow would experience if it were the only owner of a private transmission channel with a capacity equal to γ_i times that of the channel managed by the system illustrated in this invention. Λ_i therefore represents the maximum service difference with respect to an ideal situation.

A synchronous flow can therefore feature a parameter, called latency, which is calculated as follows:

$$\Theta_i = \begin{cases} \left(2 + \dfrac{\tau_i}{H_i}\right) \dfrac{N_A TTRT + \tau_{max} + \sum_{h \in S} H_i}{N_A + 1} + \tau_i - H_i, & \mathrm{se}\, H_i \ge \tau_i \\ \left(2 + \dfrac{\tau_i}{H_i}\right) \dfrac{N_A TTRT + \tau_{max} + \sum_{h \in S} H_i}{N_A + 1}, & \mathrm{se}\, H_i < \tau_i \end{cases}$$

or, for $N_A \to \infty$:

$$\Theta_i^* = \begin{cases} \left(2+\dfrac{\tau_i}{H_i}\right)TTRT + \tau_i - H_i, & \text{se } H_i \geq \tau_i \\ \left(2+\dfrac{\tau_i}{H_i}\right)TTRT, & \text{se } H_i < \tau_i \end{cases}$$

Given a switching node that implements the solution described herein, if the traffic input on a synchronous flow on that node is limited by a so-called "*leaky-bucket*" of parameters (σ,ρ), the following guarantees can be given:

a) Maximum delay on a single node for a synchronous flow

Each packet has a delay that is not greater than:

$$D = \sigma/\rho + \Theta_i$$

b) Maximum memory occupation on a node for a synchronous flow

The amount of memory occupied by packets in a synchronous flow packet is:

$$B = \sigma + \rho ? \Theta_i$$

c) Maximum delay on a route of N nodes for a synchronous flow

Let $\Phi_1 \ldots \Phi_N$ N be switching nodes that implement the system described herein; let Θ_i^j be the latencies calculated on each of the Φ_j nodes and let:

$$\overline{\Theta}_i = \sum_{j=1}^{N} \Theta_i^j$$

In this case it is possible to define an upper limit for the maximum delay for a packet to cross the N nodes, provided that the traffic input on the first node is limited by a *leaky-bucket* of parameters (σ,ρ); this limit is:

$$D_N = \sigma/\rho + \overline{\Theta}_i$$

The value $\Theta_i^* \geq \Theta_i$ can be employed in each of the three guarantees a), b), c); this means that the limits that do not depend on the number of active asynchronous flows can be calculated.

<u>Parameter selection</u>

The ability to guarantee that the synchronous flows receive a minimum service rate no lower than that requested is subordinate to a correct selection of the synchronous capacities H_i, $i=1..N_s$. Assuming that each synchronous flow i requires a minimum transmission rate r_i, it is necessary to allocate the synchronous capacities to verify the following inequality:

$$\gamma_i \geq r_i/C \qquad (3)$$

The solution described herein allocates the synchronous capacities according to two different schemes called local and global allocation respectively.

<u>Local allocation</u>

The synchronous capacities are selected as follows :

$$H_i = \frac{r_i \cdot TTRT}{C}$$

In this way, the inequality (1) is verified if the transmission rates requested verify the following inequality:

$$\sum_{h=1}^{N_S} r_h/C \leq 1-\alpha \qquad (4)$$

Each synchronous flow is guaranteed a normalised service rate equal to:

$$\gamma_i = \frac{[N_A+1]\cdot r_i/C}{N_A+\sum_{h=1}^{N_S} r_h/C+\alpha} \qquad (5)$$

The value of γ_i given by expression (5) verifies the inequality (3).

<u>Global allocation</u>

According to this scheme, which requires $N_A > 0$, the synchronous capacities are selected as follows:

$$H_i = \frac{(N_A+\alpha)\cdot r_i/C}{N_A+1-\sum_{h=1}^{N_S} r_h/C}\cdot TTRT$$

In the global allocation scheme the sum of the transmission rates requested must also remain below the inequality (4). If (4) is verified, the normalised service rate of a synchronised flow is $\gamma_i = r_i/C$.

The global scheme guarantees greater use of the channel's transmission capacity than the local scheme, in that it allocates less capacity to the synchronous flows, leaving more bandwidth for the asynchronous flow transmission.

On the other hand, the use of a global scheme means that all the synchronous capacities are to be recalculated each time the number of flows (synchronous or asynchronous) present in the system changes; the use of a local scheme, however, means that the capacities can be established independently from the number of flows in the system.

Selection of *TTRT*

The following scheme can be given to show the selection of *TTRT* in the solution according to the invention.

Given a set of synchronous flows with requested transmission rates that verify the inequality:

$$\sum_{h=1}^{N_S} r_h/C < 1$$

TTRT must be selected according to the following inequality:

$$TTRT \geq \frac{\tau_{max}}{1-\sum_{h=1}^{N_S} r_h/C}$$

The pseudo-code illustrated below analytically describes the behaviour of a system as given in the invention.

Flow initialisation

```
Sync_Flow_Init (synchronous flow i)
{     Δi=0;
      Select_synchronous_bandwidth Hi;
}
```

```
Async_Flow_Init (asynchronous flow j)
{     L_j = 0
      last_visit_time_j = start_of_curr_revolution;
}
```

<u>Visit to a generic synchronous queue i, $i = 1...N_S$, during the major cycle</u>

```
Major_Cycle_Visit (synchronous flow i)
{     Δ_i+=  H_i;
      q=first_packet_transmission_time;
      while ((Δ_i>=q)  and (q > 0))
      {      transmit_packet (q);
             Δ_i -= q;
             elapsed_time+= q;
      }
      if (q=0) Δ_i=0;
```

<u>Visit to a generic synchronous queue i, $i = 1...N_S$, during the minor cycle</u>

```
Minor_Cycle_Visit (synchronous flow i)
{     q=first_packet_transmission_time;
      if (q > 0)
      {
             transmit_packet (q);
             Δ_i -= q;
             elapsed_time += q;
      }
      if (q=0)  Δ_i=0;
}
```

<u>Visit to a generic asynchronous queue j, $j = 1...N_A$</u>

```
Async_Flow_Visit (asynchronous flow j)
{     t = current_time;
      earliness = TTRT-L_j - (t-last_visit_time_j);
      if ( earliness > 0 )
      {      L_j = 0;
             transmit_time = earliness;
             q=first_packet_transmission_time;
             while ((transmit_time>=q) and (q > 0))
             {
                    transmit_packet (q);
                    transmit_time -= q;
             }
      }
      else L_j = - earliness;
```

```
last_visit_time_i = t;
}
```

Visit sequence during a revolution

```
PTTSD revolution ()
{      elapsed_time=0;
       for (i=1 to N_S) Major_Cycle_Visit (i);
       i = 1;
       while((elapsed_time<sum(H_h)) and (i<=N_S))
       {
              if (Δ_i>0) Minor_Cycle_Visit (i);
              i ++;
       }
       for (j=1 to N_A) Async_Flow_Visit (j);
}
```

Obviously the details of how this is done can be altered with respect to what has been described, without however, leaving the context of this invention.